Master's thesis
Master's Programme in Data Science

Detecting Bat Calls from Audio Recordings

Meeri Rannisto

October 30, 2020

Supervisor(s): Arto Klami

Examiner(s): Arto Klami
Thomas Lilley

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Meeri Rannisto

Työn nimi — Arbetets titel — Title

Detecting Bat Calls from Audio Recordings

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	October 30, 2020	51

Tiivistelmä — Referat — Abstract

Bat monitoring is commonly based on audio analysis. By collecting audio recordings from large areas and analysing their content, it is possible estimate distributions of bat species and changes in them. It is easy to collect a large amount of audio recordings by leaving automatic recording units in nature and collecting them later. However, it takes a lot of time and effort to analyse these recordings. Because of that, there is a great need for automatic tools. We developed a program for detecting bat calls automatically from audio recordings. The program is designed for recordings that are collected from Finland with the AudioMoth recording device. Our method is based on a median clipping method that has previously shown promising results in the field of bird song detection. We add several modifications to the basic method in order to make it work well for our purpose. We use real-world field recordings that we have annotated to evaluate the performance of the detector and compare it to two other freely available programs (Kaleidoscope and Bat Detective). Our method showed good results and got the best F2-score in the comparison.

ACM Computing Classification System (CCS):
Hardware → Communication hardware, interfaces and storage → Signal processing systems → Digital signal processing

Avainsanat — Nyckelord — Keywords

animal sound detection, median clipping, spectral gating, bat audio analysis

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Acknowledgements

I would first like to thank my instructor for guidance. He pointed me to the right direction whenever I had trouble.

I would also like to thank my colleagues at Finnish Museum of Natural History (LUOMUS) for this thesis idea and providing me the dataset. They familiarized me with the data and provided me valuable information related to bat audio analysis.

Finally, I would like to thank my family and friends for providing me continuous support and encouragement over the years.

Contents

1. Introduction

Bats are an important taxa for biodiversity monitoring because they are sensitive to climate change and habitat loss [23]. Biodiversity monitoring allows us to see how biodiversity changes over time and use that information to develop conservation plans. It is increasingly important as many species are threatened by climate change.

Bat monitoring is commonly based on audio analysis. The analysis starts by detecting and extracting bat call segments from recordings and after that they are classified to a species level. To make it efficient, automatic tools are needed. In this work, we will present a tool for detecting the bat calls. Our method is based on a method called median clipping which has previously been used to detect bird songs [27, 28] but to our knowledge has not been used in the context of bats previously.

Audio analysis is a popular method for bat monitoring because it is a non-invasive method and it is not always easy to detect bats otherwise as they are nocturnal animals. Most bat species navigate by emitting echolocation calls and researchers have been using these calls to identify the species of the bat for a long time [12].

The bottleneck in bat monitoring is the analysis of audio recordings. One can easily get a hundreds of hours of audio recordings by leaving automatic recording units (ARUs) to the nature and collecting them later. However, it will take a lot of time to analyse them, especially if done fully manually. Because of that, automatic analysis tools are much needed.

Bat audio analysis has many challenges. Noise caused by various sources such as rain, power lines and grasshoppers can mask the calls and make them hard to detect. The calls vary by species, sex and region [32]. Their power also varies as some bats are close to the recording device and some far away. One expectation for automatic analysis tools is that they could reduce the bias that comes from human experts who have different skills [14].

In this work, we develop an automatic bat call detection program based on the median clipping method. We develop it especially for data that is collected from Finland with AudioMoth recording device. We add several modifications to the basic method in order to improve its performance. We evaluate its performance with a dataset that contains short audio recordings annotated by us. We also compare its performance with

two other bat detection programs, Kaleidoscope [3] and Bat Detective [29]. We got good results and outperformed the other programs in the comparison which suggest that this method is effective in bat call detection.

2. Background

In this chapter, we will go through basic concepts of signal processing that are needed to understand rest of the thesis. For more in depth familiarization to the topic, we recommend Giannakopoulos et al. [15].

2.1 Sampling

Recording devices are used to record a digital representation of an audio signal. A digital audio signal $x(n)$ is a sequence of discrete values, as opposed to an analog audio signal that is continuous. In practice, recording devices work by converting fluctuations in air pressure into fluctuations in electrical voltage and taking samples of that voltage periodically. The quality of a digital audio signal is strongly affected by sampling rate F_s, the number of samples taken in one second.

The Nyquist sampling theorem dictates that the sampling rate F_s needs to be at least twice as much the highest frequency we want to record. Sampling rate 192 kHz is often used to record bats since most bats use frequencies that fall below 96 kHz. Special recording devices are needed to record bats since normal devices do not need to support such high sampling rates as human hearing range is only from 20 Hz to 20 kHz [41].

Like all discrete-time signals, a digital audio signal can be represented as vectors of real numbers (see Fig. 2.1 for an example). There are as many vectors as there are channels in the signal. A digital audio signal can be a single-channel (mono) or a two-channel (stereo). Two channels are used to add a sense of multi-directional perspective to the sound. In this work, we assume that the signal has only single channel because having more channels does not provide much additional information that would be useful for our task.

2.2 Short-term processing

Audio analysis usually starts by dividing audio samples into short overlapping frames (windows). Then some useful features are extracted from the audio on a frame-by-frame

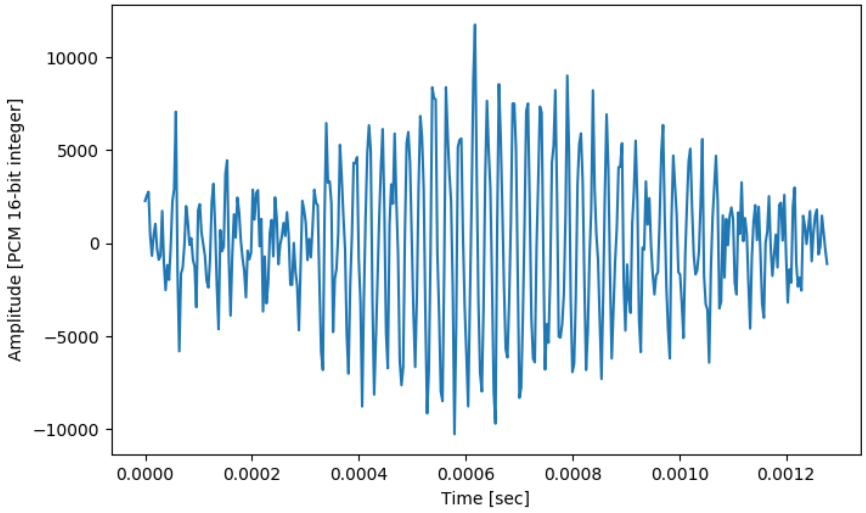

Figure 2.1: A digital audio signal. It contains one echolocation call emitted by Nathusius' pipistrelle (*Pipistrellus nathusii*).

basis. This is a called short-term processing. Short-term processing is used because audio signals are non-stationary which means that their properties change quickly over time.

When doing short-term processing, there are two important values that one has to consider. The first is window length L and the second is window step S. Window length means the number of samples in one frame. Window step or hop size determines how much there is overlap. For example, if window size is 1001 and window step is 800, the first frame will contain samples 0...1000, the second one 800...1800, the third one 1600...2600, and so on. Usually window length varies between 10-30 ms [21] or correspondingly between 1920-5760 samples if the sampling rate is 192 kHz.

Mathematically windowing is defined as follows. Let $x(n), n = 0, ..., N-1$ be an audio signal of length N and $w(n)$ window function. Window function is a function that is zero outside a finite interval. For example, a rectangle window function is defined as

$$w(n) = \begin{cases} 1, 0 \leq n \leq L-1 \\ 0, \text{otherwise} \end{cases} \tag{2.1}$$

Then we can calculate the windowed signal $x_i(n)$ by equation

$$x_i(n) = x(n)w(n - i * S), i = 0, ..., M-1 \tag{2.2}$$

where M is the number of frames. From that equation, we can see that frame i contains samples $i * S, ..., i * S + L - 1$ and derive $M = \lfloor \frac{N-L}{S} \rfloor + 1$.

It is possible to use a different window function than the rectangular. Other commonly used ones are the Hann window and the Hamming window (see Fig. 2.2). They

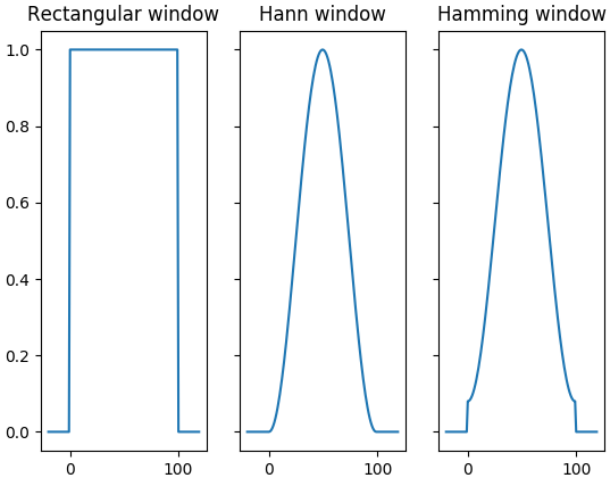

Figure 2.2: Rectangular, Hann and Hamming window when window size is 100.

are smoother functions than the rectangular. They are usually used when calculating Short-Time Fourier Transform (see Section 2.3) to reduce the effect of the discontinuity between the signal amplitudes at the boundaries of the window.

In addition to short-term, sometimes mid-term analysis is used. Mid-term analysis divides the audio to bigger segments than short-term. The segment length is commonly between 1-10 seconds [21]. For each segment, short-term analysis is carried out and some useful statistics such as mean or variance are calculated from the short-term features.

2.3 Frequency spectrum

When the signal is in time domain, one can easily see how the amplitude varies over time, but it is difficult to say anything about frequencies. To see how the amplitude varies over frequencies, the signal is converted to a frequency-domain representation (also called frequency spectrum or spectral representation). Frequency domain representation often gives more information about the content of the audio than the time-domain representation.

Conversion from time-domain to frequency-domain can be done using the Discrete Fourier Transform (DFT). In practice, it is done using the Fast Fourier Transform (FFT) which is a computationally efficient version of DFT. DFT is reversible so it is possible to reconstruct the original signal using the inverse DFT (IDFT) or inverse FFT (IFFT).

Let $x(n), n = 0, ..., N-1$ be the sequence of audio samples. Then DFT is calculated by equation

$$X(k) = \sum_{n=0}^{N-1} \exp(-\frac{i2\pi}{N} kn), k = 0, ..., N-1 \tag{2.3}$$

where i is the imaginary unit, defined by $i^2 = -1$. The result is a sequence of complex numbers of length N called DFT coefficients. The kth coefficient corresponds to a frequency f_k, $f_k = k\frac{F_s}{N}$.

We can convert DFT coefficients to real numbers by taking their magnitude (absolute value). Magnitude of the DFT is called magnitude spectrum. A magnitude of the kth coefficient, $|X(k)|$, represents the intensity of the audio signal at frequency f_k. Magnitude spectrum has the property that it is always symmetrical around point $k = \lceil\frac{N-1}{2}\rceil$ when the signal $x(n)$ is real valued. Because of that, only the first $\lceil\frac{N-1}{2}\rceil$ coefficients are needed, that is $k = 0, ..., \lceil\frac{N-1}{2}\rceil$.

As mentioned in Section 2.2, audio is usually analysed in short-term. The same applies when converting the signal from time-domain to frequency-domain. The signal is first divided into short overlapping frames. Then each frame is converted to frequency-domain with the DFT. This is called Short-Time Fourier Transform (STFT). The window size affects the frequency and time resolution of the STFT. When the window is wide, STFT has high frequency resolution but low time resolution. When the window is narrow, STFT has high time resolution but low frequency resolution. One has to decide the best trade-off between them.

Beside the window size L, the window function $w(n)$ and the window step S also affect the quality of the STFT. Smooth window functions such as the Hamming reduce an effect called spectral leakage which causes the STFT to appear as more blurred. Spectral leakage is due to the fact that the DFT assumes that the signal is one period of an infinite periodic signal and that assumption is not met. A downside of smooth window functions is that some data may be lost as they set values close the window boundaries near to zero. However, this can be countered by setting the window step low enough so that there will be enough overlap between the frames.

Audio signals are often inspected visually using a spectrogram (sometimes called sonogram). The spectrogram shows how frequency spectrum varies over time, see Fig. 2.3 for an example. Officially, a spectrogram is defined as squared magnitude of STFT, $|\text{STFT}|^2$. In practice, however, the term is used more broadly to mean any visual representation of audio obtained by STFT.

2.4 Noise in audio

Noise means all the other sounds in the audio signal besides the sounds that we are interested in. In bat sound detection, all other sounds besides bat sounds are noise. There are many different kinds of noise. They can be divided into three types based on their source: biophony, geophony and anthropophony [25]. Biophony sounds refer to those emitted by living organisms such as birds and grasshoppers. Geophony sounds are naturally occur-

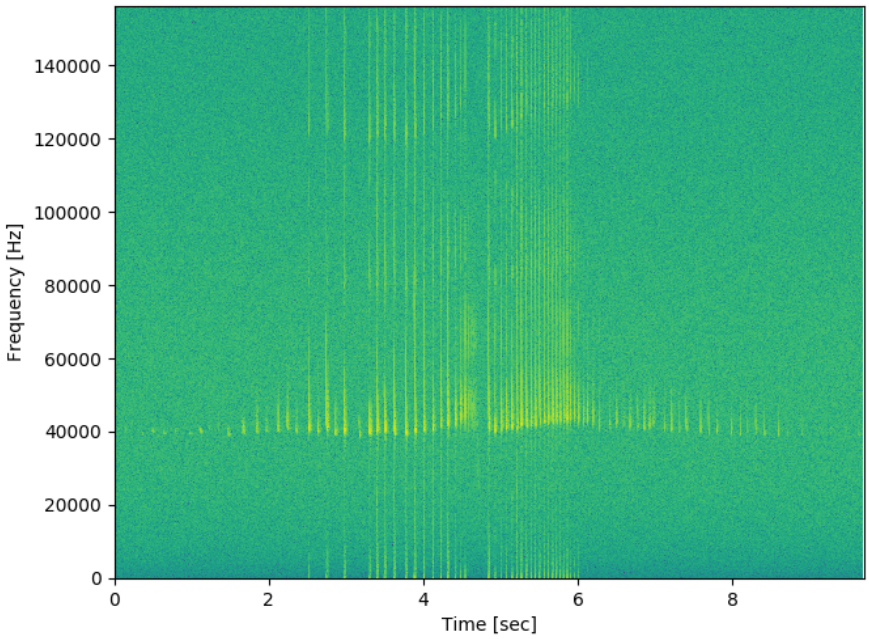

Figure 2.3: Spectrogram that contains echolocation calls emitted by Nathusius' pipistrelle (*Pipistrellus nathusii*). It is from the same recording that is depicted in Fig. 2.1.

ring non-biological sounds such as rain or wind sound. Anthropophony sounds means all sounds produced by humans or human-made devices such as the recording device.

Noise is often assumed to be additive and it is modeled with additive noise model. In that model, noisy signal $y(t)$ is a sum of clean signal $x(t)$ and additive noise $n(t)$, that is

$$y(t) = x(t) + n(t), \tag{2.4}$$

where t is time index. The aim of a noise reduction algorithm is to estimate signal $x(t)$ given the noisy signal $y(t)$. We will cover noise reduction algorithms more in Section 3.1. One classic one is called spectral subtraction (see Section 3.1.2), which subtracts an estimated noise spectrum from a noisy signal spectrum to get a clean signal spectrum.

2.5 Audio features

We can extract simple features from the short-term audio segments and use them for audio event detection (covered in Section 3.2.1). These features can be divided into three classes: *time-domain features*, *spectral features* and *perceptual features*. Time-domain features such as *energy* and *zero-crossing rate* are computed directly from the signal. Spectral features such as *spectral flatness* and *spectral flux* are computed from the frequency spectrum of the signal. Perceptual features such as *loudness* and *pitch* try to match the human perception of sound. Next we will describe some of the most common features in more

detail, focusing on features that we consider promising for bat call detection.

2.5.1 Time-domain features

Energy

Energy is the most common audio feature. It represents amplitude variation over time. Let $x_i(n), n = 0, ..., N-1$ be the sequence of audio samples in the ith frame and N the length of the frame. Then the energy is calculated by equation

$$E(i) = \sum_{n=0}^{N-1} |x_i(n)|^2. \tag{2.5}$$

Energy is often divided by the frame length in order to make it independent of the frame length. The official name for that measure is 'power' but the term 'energy' is also often used. The equation is

$$P(i) = \frac{1}{N} \sum_{n=0}^{N-1} |x_i(n)|^2. \tag{2.6}$$

Energy can be used to detect silent regions for example as they have low energy.

Zero-crossing rate

Zero-crossing rate [43] is a rate of sign-changes in the signal. A sign-change happens when the signal values go from negative to positive or the other way around. A formal definition for zero-crossing rate is

$$\text{ZCR}(i) = \frac{1}{N-1} \sum_{n=1}^{N-1} \frac{|\text{sgn}(x_i(n)) - \text{sgn}(x_i(n)-1)|}{2}$$

, where

$$\text{sgn}(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases}.$$

Zero-crossing rate can be used for example to separate unvoiced and voiced speech [40]. Unvoiced parts usually have higher zero-crossing rates than voiced parts.

2.5.2 Spectral features

Spectral flux

Spectral flux [44] represent how quickly the spectrum of the signal changes. It is calculated as the squared difference of two consecutive STFT frames. Let $X_i(k), k = 0, ..., K-1$ be the magnitude spectrum of ith frame. Then spectral flux is calculated as

$$\text{Flux}(i) = \sum_{k=0}^{K-1} |X_i(k) - X_{i-1}(k)|^2.$$

Usually normalized versions of DFT coefficients are used when calculating spectral flux. With some signals, the spectrum changes slowly and with some signals, it changes quickly. Spectral flux can be used to separate those two signal types.

Spectral flatness

Spectral flatness [22] represent how tonal the signal is and it can be used to separate tone-like signals from noise-like. It is a geometric mean of the spectrum divided by its arithmetic mean:

$$\text{Flatness}(i) = \frac{\sqrt[K]{\prod_{k=0}^{K-1} X_i(k)}}{\frac{1}{K}\sum_{k=0}^{K-1} X_i(k)}.$$

High spectral flatness indicates that the power is divided evenly across frequency bands and it resembles white noise. Low spectral flatness indicates that it is concentrated on small number of bands.

2.6 Listening to bat calls

Bat calls can be listened to by lowering the playback speed. If a recording is played at 0.1 playback speed, the speed and the signal frequency is ten times lower than in the original recording. With a slow playback speed, bat calls sound a little similar to the sound of a bird.

Lowering the playback speed has the same effect than setting the sampling rate F_s lower than it really is. It is called time expansion. Figure 2.4 shows a spectrogram of a time-expanded signal where the sampling rate is set $\frac{1}{10}$th of the original sampling rate.

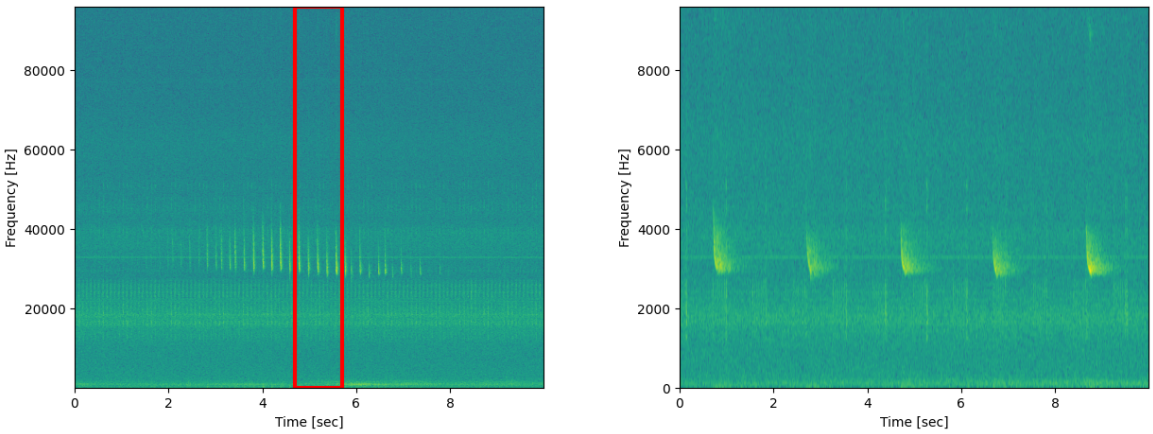

Figure 2.4: Left is a normal spectrogram. Right is is a spectrogram of a time-expanded signal from a part that is highlighted with red rectangle in the left spectrogram.

2.7 Image processing

Some audio event detection methods such as the median clipping (see Section 3.2.3) are based on detecting the target signal from the spectrogram using image processing techniques. The aim is to detect big objects or shapes that look like the signal. In the median clipping method, several common image processing techniques can be utilized: morphological operations, blurring and connected-component labeling. We will cover those in this section. Before that, we take a look at common noise types that are present in the image.

2.7.1 Different noise types

Digital images are always noisy. The aim of image processing is often to reduce that noise. Different image processing methods work for different types of noise. Two common noise types are Gaussian noise and salt-and-pepper noise. They are depicted in Fig. 2.5 by Myllykoski [33].

Gaussian noise means noise that values are Gaussian-distributed. In other words, they follow the distribution:

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(z-\mu)^2}{2\sigma^2}},$$

where z is the noise value, μ is the mean and σ the standard deviation. White noise is special type of Gaussian noise where the values are independent and identically distributed. Gaussian noise can be seen in almost all spectrograms, it makes the background of the signal unevenly colored (see Fig. 2.3 for example).

Salt-and-pepper noise, also called impulsive noise, means the type of noise where some of the pixels are corrupted high level noise. In a black-and-white image, it appears as single white pixels inside black objects and single black pixels inside white objects.

Figure 2.5: Left image is corrupted by salt-and-pepper noise, right image by Gaussian noise [33].

2.7.2 Morphological operations

Morphological operations [17] transform the image based on the shapes of the objects in the image. Morphological operations are usually performed on a binary image but it is possible to use them for grayscale images as well. In this section, we will assume that the image is binary.

Morphological operations use a structuring element (kernel) to define a 'neighborhood' for each pixel in the image. A structuring element is usually a small binary matrix where value 1 means that the pixel is included in the neighborhood and 0 that is it not. It has commonly odd dimensions so that its origin can be defined to be at its centre. It has two properties: shape and size. For example, a 3x3 box-shaped structuring element (see Fig. 2.6) defines that the neighborhood for each pixel is all eight pixels around it and itself.

Morphological operations are based on mathematical morphology (MM). It is a theory for processing and analysing geometrical structures and is based on set theory. Let us remember that $\mathbb{Z}^2$ means the set of all pairs of integers, $\mathbb{Z}^2 = \{(x, y) : x \in \mathbb{Z}, y \in \mathbb{Z}\}$. In MM, a binary image $A \subset \mathbb{Z}^2$ and a structuring element $B \subset \mathbb{Z}^2$ are defined by the coordinates of pixels with value 1. For example the cross in Fig. 2.6 would be given by $B = \{(-1, 0), (0, -1), (0, 0), (0, 1), (1, 0)\}$. These notations are used when the operations are defined in the following sections.

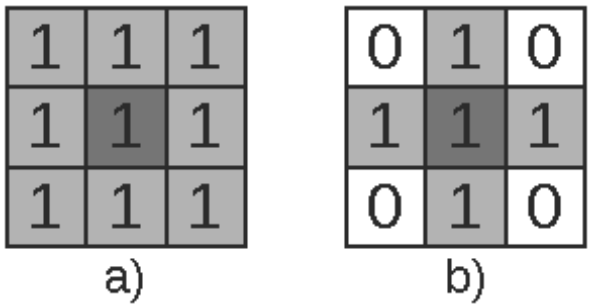

Figure 2.6: Two common structuring elements of size 3x3 that are shaped as box (a) and cross (b).

Dilation and erosion

Dilation and erosion are the most basic morphological operations. All other operations are based on them. Dilation expands the objects in the image, whereas erosion shrinks them. Dilation sets to 1 all pixels that have at least one pixel in their neighborhood that is 1. Erosion sets to 0 all pixels that have at least one pixel in their neighborhood that is not 1. See Fig. 2.7 how the effect of these operations look in practice when applied to a spectrogram that is converted to a binary image.

The effect of these operations depends on the structuring element. For example,

a dilation with a 3x3 box-shaped structuring element would expand objects little in all directions, whereas a 11x1 rectangular-shaped structuring element would expand objects only in horizontal direction.

Mathematically these operations are defined as follows. Let the image be $A \subset \mathbb{Z}^2$ and the structuring element $B \subset \mathbb{Z}^2$. Translation of A by vector $x \in \mathbb{Z}^2$ is defined as

$$A_x = \{c : c = a + x, \text{ for } a \in A\}.$$

Then, dilation is defined:

$$A \oplus B = \bigcup_{b \in B} A_b$$

and erosion is defined:

$$A \ominus B = \{z \in \mathbb{Z}^2 | B_z \subseteq A\}.$$

Closing

Closing is dilation followed by erosion. The same structuring element is used for both operations. It can be used remove small holes in the objects and pepper-noise (small regions of zeros in the middle of ones). Closing is defined

$$A \bullet B = (A \oplus B) \ominus B.$$

Opening

Opening is erosion followed by dilation. The same structuring element is used for both operations. It can be used to remove small-objects and salt-noise (small regions of ones in the middle of zeros). Opening is defined

$$A \circ B = (A \ominus B) \oplus B.$$

2.7.3 Blurring

Blurring is a common image processing method that can be used for all kinds of images, binary, grayscale or colored.

Gaussian blur

Gaussin blur convolves the image with a kernel that values corresponds to two dimensional Gaussian distribution:

$$g(\boldsymbol{z}) = \frac{1}{\sqrt{(2\pi)^2 |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{z}-\boldsymbol{\mu})},$$

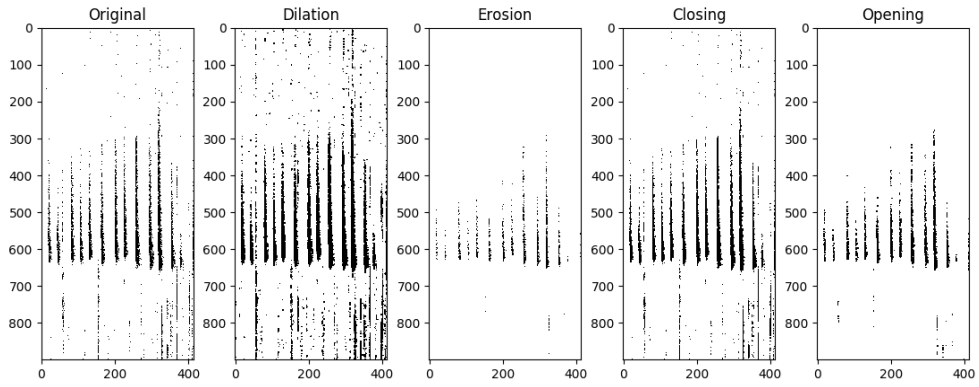

Figure 2.7: The effect of morphological operations to a spectrogram that is converted to a binary image. A 3x3 box was used as a structuring element.

where a column vector $\boldsymbol{z}$ denotes coordinates of a point. Gaussian blur can be used to remove Gaussian noise and blur the objects in a image. The effect of the blur depends on the size of the kernel, the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$.

Median blur

Median blur sets every pixel to a median of values in its neighborhood. Neighborhood is defined by a kernel similarly than in morphological operations. Median blur is especially useful for removing salt-and-pepper noise. It is better than Gaussian blur at preserving the edges of the objects in the image.

2.7.4 Connected-component labeling

With connected-component labeling [19], it is possible to find connected areas in a binary image. It is useful for many purposes, for example it can be used to filter small components out of the image and keep only the big ones. Connected-component labeling is an application of graph theory.

For undirected graphs, a connected component is a subgraph in which any two nodes are connected by a path. None of the nodes in the subgraph are connected to any nodes that are not in the subgraph. There are many algorithms for finding these connected components. The same algorithms can be used to find the connected areas of an image if the image is first converted into a graph.

A binary image can be easily converted to a graph. All the pixels with value 1 are considered nodes in the graph. All the pixels that are neighbors in the image have an edge between them in the graph. There are several ways to define the neighborhood. 8-connectivity means that all the pixels that touch an edge or a corner of a pixel are its neighbors. 4-connectivity means that all the pixels touch a edge of a pixel are its

neighbors.

2.8 Classification

Bat sound detection is a binary classification problem. The goal is to classify audio segments into two classes: those that contain bat sounds (either calls or individual pulses) (class 1) and those that do not (class 0). There are many possibilities to what the used audio segments can be. They can be short-term frames (see Section 2.2) or longer segments of a length 10 seconds, for example. It is also possible to classify each individual pixel in a spectrogram to get both frequency and time information. In this work, we are going to classify for each short-term frame whether or not it is part of a bat call.

Supervised machine learning is commonly applied to classification problems. Supervised machine learning uses manually labeled training data to predict the labels (classification) of new data. For bat sound detection, training data is a set of audio signals with labels for each part in that signal. Formally, it is a set of length N and it can be denoted as $\{(x_1, y_1), ..., (x_N, y_N)\}$ where x_i is an audio segment and y_i its class. The goal of supervised learning is to find a function $f(x)$ that fits the training data best based on some scoring measure.

It is not always possible to use supervised learning as the collecting and labeling training data takes a lot of work. Especially in the field of animal sound detection, there are not yet many publicly available sound repositories and making one on one's own is often unfeasible. Because of that, one might want to use other kind of algorithms that do not rely on supervised learning but instead attempt to detect bat sounds based on some simple feature such as their shape in the spectrogram. In Section 3.2, we take a closer look at algorithms that can be used for this problem.

2.8.1 Performance measures

For the performance evaluation of a classifier, there needs to be a manually labeled testing dataset. The labels the classifier gives to that dataset are compared with the human-made labels. Testing dataset has to be different than the training data to get more objective results. Classifiers tend to overfit to the training data which means they give better results with it than with some new unseen data. For that reason, it is important to use data that is not used in training for testing.

Accuracy, Precision, Recall

Many performance measures are based on a confusion matrix (See Table 2.1). Four values are calculated: true positives, true negatives, false positives and false negatives.

True positive is a sample that is correctly labeled as 1 and true negative is a sample that is correctly labeled as 0. False positive is a sample that is incorrectly labeled as 1 and false negative is a sample that is incorrectly labeled as 0. From these values, three common performance measures can be calculated: accuracy, precision and recall.

	Actual positive	Actual negative
Predicted positive	TP (True Positive)	FP (False Positive)
Predicted negative	FN (False Negative)	TN (True Negative)

Table 2.1: Confusion matrix

Accuracy is the most common performance measure. It is the number of correct predictions divided by total number of predictions:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

It usually measures well the overall performance of a classifier but not in the case where one class is highly more prevalent than the other. In bat sound detection, class 0 (contains no bat) is highly more prevalent than class 1 (contains a bat) and accuracy is not a very good performance measure.

Precision is the number of samples correctly predicted as 1 divided by the total number of samples predicted as 1:

$$\text{precision} = \frac{TP}{TP + FP}$$

Recall is the number of samples correctly predicted as 1 divided by the total number of samples actually belonging to class 1:

$$\text{recall} = \frac{TP}{TP + FN}$$

It depends on the application which of these is the more important performance measure. When it is important that most of the samples that are predicted as 1 actually belong to class 1, then precision is more important. When it is important that most of the samples that belong to class 1 are predicted as 1, then recall is more important.

F-score

F-score takes account both precision and recall. It is calculated as

$$F_\beta = (1 + \beta^2) * \frac{\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) * \text{recall}},$$

where β is a positive number. The bigger β is, the more important recall is considered to be compared to precision. F_1 is the harmonic mean of precision and recall.

Other measures

In the field of voice activity detection (VAD), performance is commonly measured by four parameters: FEC (Front End Clipping), MSC (Mid Speech Clipping), OVER and NDS (Noise Detected as Speech) [13]. FEC is speech that is detected as noise at the beginning of an utterance. MSC is speech that is detected as noise in the middle of an utterance. OVER means noise that is detected as speech after the end of an utterance. NDS refers to noise in other places that is detected as speech. The parameters are asymmetrical as clipping at the beginning of an utterance has its own parameter but clipping at the end has not. They are designed for algorithms that have a delay in switching between the two states (speech and noise). Because of that, these performance measures did not seem very useful for our method which works in a different manner.

In addition to objective measures, subjective tests have also been utilized in VAD evaluation [7]. In a subjective test, users listen to processed recordings and give evaluations on some measures such as the quality of speech and comprehension difficulty. We did not use any subjective tests in this work as we concluded they would not give a meaningful amount of new information compared to the objective ones.

3. Audio event detection

Animal sound detection and extraction is an important part of the automatic identification process. Some researchers have argued that it is also the most challenging part [45]. Many published works in the area focus on species identification and use manually or semi-automatically segmented data [36]. Nevertheless, the interest in animal sound detection is growing, for example the first bird audio detection challenge was held in 2016 [46]. In this chapter, we take a look at existing approaches.

Animal sound detection is closely related to voice activity detection (VAD) [13, 5, 47] and audio event detection [9]. Voice activity detection is similar to animal sound detection but instead of detecting animal sounds, human speech is detected. Audio event detection (also called acoustic event detection or sound event detection) is a general term for detecting all kinds of audio events. In this chapter, we will concentrate on methods that have been used for animal sound detection but also cover briefly methods from these closely related fields.

Audio event detection usually has three steps: (1) noise reduction, (2) feature extraction and (3) classification. We will start this chapter by looking into noise reduction methods (step 1) and then detection methods (steps 2 and 3).

3.1 Noise reduction

The aim of noise reduction is to reduce noise in the audio and make the target sounds more easy to detect. The performance of a detector depends a lot on the noise reduction as noise can generate a lot of false positives and mask target sounds. There are many noise reduction methods, such as filtering [48] and spectral subtraction [8]. Different methods work for different type of noise.

3.1.1 Filtering

Filters can be used to reduce sounds with frequencies that are outside the target frequency range. A filter that reduces sounds above some predefined cutoff frequency is called a low-pass filter and a filter that reduces sounds below it is called a high-pass filter. Band-pass

filter is a combination of these two. In the case of bat sound detection, a high-pass filter is often used. As most noise is on the low frequencies and bat calls only on the high frequencies, a high-pass filter works well. For example, Henríquez et al. applied a 10th order high-pass Butterworth filter with cutoff frequency 28.8 kHz [20].

In signal processing, a method called the Wiener filter [48] is commonly used for noise reduction. However, the method assumes that both the signal and noise are stationary which does not hold true for animal sounds. It also assumes that the spectrum of the noise is known. That often does not hold true either although it is possible to estimate the noise spectrum from the noisy signal as discussed in next section.

3.1.2 Spectral subtraction

Spectral subtraction [8] is a traditional noise reduction algorithm. It can be used to remove stationary noise such as white noise. It estimates the spectra of a clean signal by subtracting the average noise spectrum from the noisy signal spectra. After that, the clean signal can be constructed from the clean spectra with the inverse STFT.

The method assumes that the noise is stationary and that its spectrum is known. The noise spectrum can be estimated from a part of audio that is known to only contain noise. Quite often it is assumed that the first few frames do not contain anything else than noise and they are used for estimation. In many real life applications, however, this assumption cannot be done. Another alternative is to assume that low-energy sections contain only noise.

3.1.3 Spectral noise gating

Spectral noise gating [24, 42] is another spectral noise removal method. In general, a noise gate means a device or software that attenuates a signal that is below a threshold by a fixed amount. A signal that is above the threshold is preserved. In spectral noise gating, the operation is applied to the STFT of the signal so that each frequency band has its own threshold.

The method starts by estimation of the threshold. First, noise mean $\boldsymbol{\mu}_n$ and standard deviation $\boldsymbol{\sigma}_n$ are estimated from a part of audio that is known to only contain noise. Then for each frequency band k, a threshold $\boldsymbol{\theta}(k)$ can for example be:

$$\boldsymbol{\theta}(k) = \boldsymbol{\mu}_n(k) * \boldsymbol{\sigma}_n(k) * \alpha,$$

where α is a sensitivity value. The greater the sensitivity is, the more values are considered noise.

After the threshold is estimated, a mask is computed. All pixels that are below the threshold in the spectra get the value 1 and all others get the value 0. The mask is usually

smoothed over both frequency and time. After that, a clean spectra can be obtained by subtracting the mask multiplied by some fixed value from the noisy spectra.

3.1.4 Wavelet denoising

Wavelet denoising [16] is a slightly more recent noise reduction method than the ones presented earlier. It is a general method that can be applied to images as well as to audio signals. In the field of bioacoustics, it has been successfully used to reduce noise from bird song recordings [37].

Wavelet denoising is based on wavelet transform. Wavelet transform is an alternative to the Fourier transform (see Section 2.3). Wavelet transform is defined by a basis function called *mother wavelet* which is a function that satisfies certain mathematical properties. If the transform is reversible, the original signal can be obtained with inverse wavelet transform.

Wavelet denoising consists of three steps. First the input signal is transformed to a wavelet transform domain. Then the wavelet transform is denoised. The denoising is based on the fact that the noise tends to be have small values in the wavelet domain and those values can be shrinked or removed without affecting the target signal. After that, the signal can be reconstructed using the inverse wavelet transform.

The advantage of this method is that it is enough for the noise to be nearly stationary for this method to work. The disadvantage is a high computational cost. There is also a challenge in finding the best parameters (mother wavelet, decomposition level and denoising threshold) that work for all kinds of audio files.

3.2 Detection Methods

There are several different methods that can be utilized to detect audio events. They can be divided into four categories: feature-based methods, template matching, median clipping method and supervised methods. Feature-based methods and template matching are traditional methods that has been used for animal sound detection for a long time [11, 6]. Median clipping method is a newer method that was developed for NIPS4B 2013 Bird Challenge by Lasseck [27]. Supervised methods are methods that use training data to predict the classifications of new data and they are quite popular in recent years [5, 29].

3.2.1 Feature-based methods

Feature based methods extract one or more features (See Section 2.5) from the audio. Then some threshold is used to classify the frames to one of the two classes (has target sound or has not). See Fig. 3.1 for example. Features such as energy [20, 50], zero crossing

rate (ZCR) [4], signal envelope [34, 35], MFCCs, perceptual or spectral features [47] and wavelet [45] have been used.

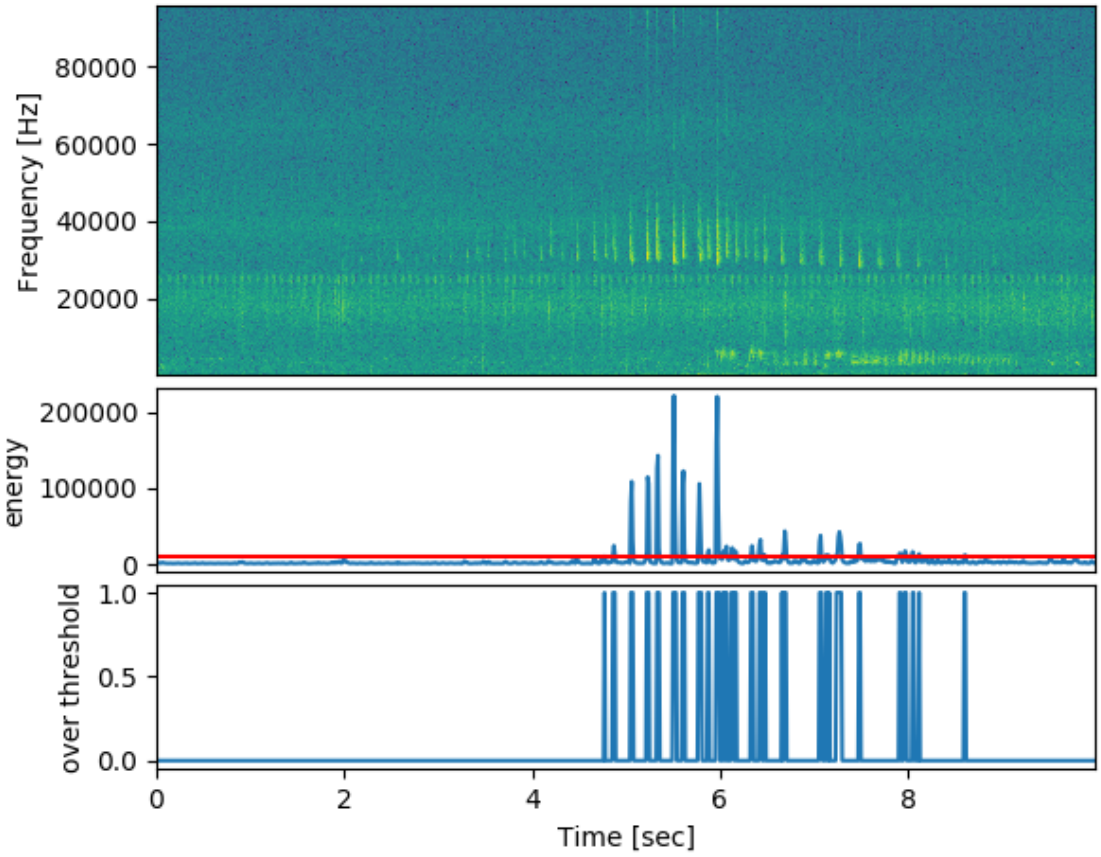

Figure 3.1: Example of a very simple energy-based method. First energy is calculated for each frame (middle plot). Then some threshold is decided, in this case a fixed value is used (shown as red line). Finally, all the frames with greater energy than the threshold are classified to class 1 (bottom plot).

Features

Signal energy is a simple feature that was already used in the early VAD algorithms [39] and is still commonly used. It is often used together with some other feature such as the ZCR or some spectral feature. It is often normalized in some way, for example by estimating the signal-to-noise ratio. Using the energy is based on the assumption that the parts that have human or animal sounds also have more energy.

Signal envelope is a similar feature than the energy. The difference is that it is not a short-term feature but a continuous curve that traces the peaks of the signal. It can be obtained with an operation called Hilbert transform.

Perceptual features such as the pitch and the harmonicity are commonly used in VAD. Human speech can be divided into voiced and unvoiced parts. Voiced parts usually

have a distinct pitch and are harmonically rich unlike most of the noise. Same applies to many other animal sounds such as bird songs. These features might not be so suitable for detecting bat calls as the calls are rather pure tones and not so harmonically rich except the social calls [31]. Also, one notable downside of these features is their high computational cost.

There are several spectral features that can be used. Spectral features are not so computationally intensive as perceptual features but in some cases can be used for same purposes. For example, Tan et al. [47] have noted that low spectral flatness seem to indicate a distinct pitch.

Wavelet coefficients are not only useful for noise reduction (see Section 3.1.4), they can also be used for audio event detection. Several features can be computed from these coefficients. For example, Selin et al. [45] used maximum energy, position, spread and width for bird sound detection with promising results. One challenge in using wavelet features is how to find the best mother wavelet and decomposition level which is the same as in wavelet denoising.

Threshold and hangover scheme

Threshold is either fixed or adaptive. Fixed threshold stays same for whole audio file whereas adaptive threshold is updated when the estimated noise level changes. Adaptive thresholds are more robust against changing noise.

A statistical model is an another way to find suitable threshold. Statistical model based methods estimate the probability distribution of features for target sound and noise. Then likelihood ratio is computed for each frame and a fixed threshold is applied. For example, Alam et al. [5] assumed that 10% of frames with the highest energy contain only speech and 10% of frames with the lowest energy contain only noisy and fitted Gaussian mixture models to features extracted from them.

Feature-based methods usually use some kind of hangover scheme to get trailing parts of the speech or call. Trailing parts are usually fainter than other parts of human speech. A hangover scheme slows the transition from speech to non-speech and smooths the decision boundaries.

Advantages and challenges

Feature-based methods are often simple and computationally non-intensive methods. They are good methods when it is important that the detection algorithm does not take too much time. They are also relatively easy to implement and do not require any training data or templates.

The main challenge in using a feature-based method is finding the best feature and

threshold. A lot of research has been done in finding the best features for detecting human speech but not so much for detecting animal sounds. Energy is a popular feature but it is based on the assumption that animal sounds would have more energy than noise which is often not the case [36]. For example, rain drops that hit close to the recording device can have much more energy than most of the animal sounds.

3.2.2 Template matching

Template matching method needs one or more template for each species with a distinct call. Template is a two-dimensional image, either a full spectrogram of the target call or a frequency contour of it. The template is cross-correlated with a spectrogram of a recording. The points where the cross-correlation is high enough are assumed to contain the target call. Cross-correlation that is calculated can either be one-dimensional or two-dimensional. One-dimensional cross-correlation is significantly faster than two-dimensional but does not work with calls with a different frequency.

Template can be a spectrogram of a real call or it can be constructed synthetically from many calls [30]. The aim of a synthetic template is to construct an "average call" that would be a better template than any single real call.

Template matching seem to perform well in practice. A commercial bat call detection program SCANR [1] is said to use template matching [18] and it has achieved good results, detecting over 99% of the echolocation calls in one test [49]. The downside of this method is the high computational cost. Cross-correlation, especially two-dimensional, is computationally intensive. Another problem is that it is a quite species-specific method. There needs to a template for each species unless the calls of different species are very similar to each other.

3.2.3 Median clipping method

Median clipping method extracts a binary image from a spectrogram and uses standard image processing techniques to find regions of interests (ROIs) from it. Median clipping method has been previously used to segment short audio clips containing bird sounds [27, 28, 10].

First a spectrogram is computed. Then Gaussian blur can be applied to it for noise reduction [10]. After that, it is converted to a binary image with median clipping. Median clipping sets all pixels that are more than n times median of its column and more than n times median of its row to one and others to zero. Commonly $n = 3$ is used. Then standard image processing techniques are used, for example closing, dilation and median filter [27, 28, 10]. After that, all connected pixels are defined as a segments and all segments that are big enough are labeled as ROI.

One advantage of a median clipping method is that it combines the noise reduction and segmentation. Median clipping ignores broadband and continuous noise to a certain extent and image processing reduces white and salt-and-pepper noise. Another advantage is that the method is general and not species specific. It also seem to perform quite well. However, it may have a problem with loud background noise or with faint calls.

3.2.4 Supervised methods

Various supervised methods such as a neural network [29] and a supervised Gaussian mixture model [5] can be used for animal sound detection. The features that are given as input can be more complex than in unsupervised methods mentioned in Section 3.2.1. For example, the method can be convolutional neural network (CNN) and the input a spectrogram of the audio [29].

Collecting a large training dataset is a laborious work and the main challenge in using supervised methods. A large number of people may be need to be involved to make it possible to construct such a dataset. One option is to start a citizen science project and collect annotations from public users as Mac Aodha et al. did in their work [29].

Supervised methods have many advantages. There is no need to define so precisely which features and thresholds the algorithm should use for classification but they are learned directly from the training data. They usually have good performance. The main challenge in supervised methods is the need for large manually labeled training dataset. Also, these methods do not work well if the audio contains unexpected sounds that are not featured enough in the training data.

4. Materials and methods

4.1 Materials

Our dataset consists of 139 ten minutes long audio recordings that were randomly sampled from a large dataset of roughly 25 000 recording. Only a small subset was used because the recordings did not have any annotations and it would have not been feasible to annotate them all in this work. The dataset was used to develop the detector and test its performance.

The dataset was collected as part of the "Lukiolaiset lepakkotutkijoina" ("High-schoolers as bat researchers") science project [2]. The project was organized by University of Helsinki, Finnish museum of natural history (Luomus) and Natural Resources Institute Finland (Luke). The goal of the project was to collect bat sound recordings from all over Finland in the summer of 2019. 180 small groups of high schoolers participated to the project and each of them collected recordings using the AudioMoth recording device (Fig. 4.1).

Recordings were made during weekends every two weeks. In total, there were nine recording weekends. The recording devices were programmed to make ten minutes long recordings at fixed time points through every recording night. High schoolers were instructed to place the recording devices in places where they believed the bats could be. Places that are near watercourses, yards, parks and sparse forest areas were given as recommendations. Places that did not have much human traffic, strong background noise or electric fences or power lines were preferred so that there would not be so much disturbances in the recording. Each recording device was placed at the same location each recording weekend.

Sample rate 192 KHz was used, so that all sounds up to 96 kHz were recorded (see Section 2.1). The audio was recorded to 16-bit single-channel WAV format.

4.1.1 Data sampling and annotation

From the whole dataset, 139 recordings were randomly chosen so that each of them was recorded with a different device. After that, they were randomly divided into two groups

Figure 4.1: Left: Picture of the Northern bat (*Eptesicus nilssonii*). Right: Picture of AudioMoth recording device. [2]

of sizes 69 and 70. The first group was used to develop the detector and fine-tune its parameters and we are going call it "development set" in this paper. The second group was used to test the performance of the detector and we are going to call it "testing set".

All the recordings in the development and testing set were annotated by the author. To make the annotation process easier, we created a program (Fig 4.2). The program allows the user to see the spectrogram of the audio and listen it with 0.1 playback speed where bat calls can be heard (see Section 2.6). Annotations are made by creating a red rectangle and dragging it so that the left side of the rectangle points to the start of a bat call and the right side to the end of the call. The annotations were saved to a text file as a list of time ranges in seconds.

We did the annotations mostly by only looking at the spectrogram of the recording and not listening to it. When there was anything that resembled bat calls, we listened that part. Listening the audio made it easier to determine the starting and ending points of the calls as the pulses are usually very faint at the ends and not visible on the spectrogram. The annotations would be more reliable if all the recordings were listened through but it was not feasible in this work. It would take too much time especially if 0.1 playback speed was used.

In addition to listening the audio more, the reliability of the annotations would improve if they were done by an expert and by multiple people. The calls are relatively easy for non-expert to recognize because there are not many other sounds on high frequencies. Faint calls and calls overlapping with other sounds can still be challenging. Having multiple people do the annotations and aggregating results would mitigate biases the individual annotators have.

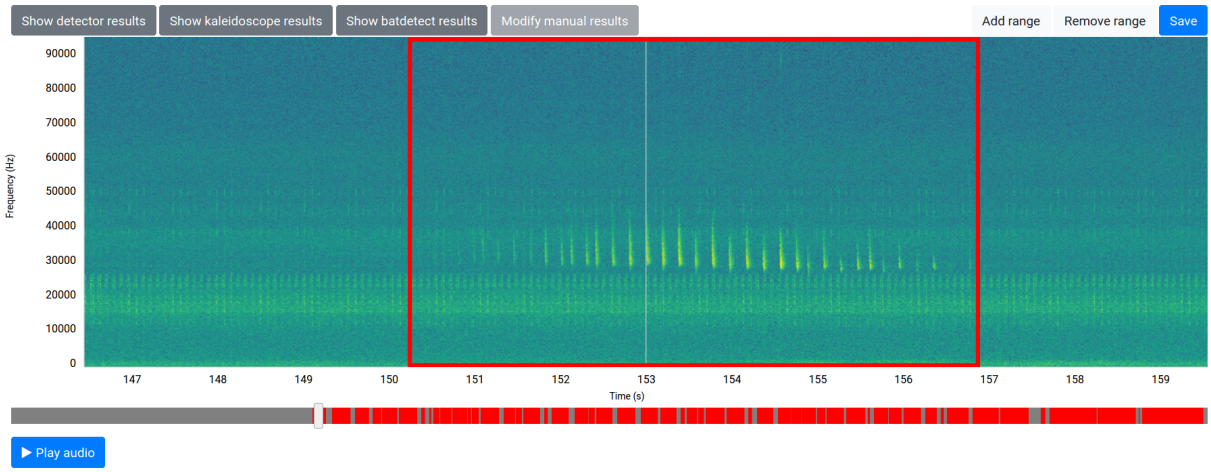

Figure 4.2: The user interface of the program that was used to annotate the audio files. The left and right side of the red rectangle shows the start and end of the bat call.

4.1.2 Audio content

Of the 139 files annotated, 27 contain one or more bat calls (about 19%). In minutes, 24 out of a total of 1390 minutes (about 2%) contain bat calls. The numbers are relatively high which indicates that the method for collecting recordings is efficient for capturing bat calls. These numbers also show how the size of the dataset could be reduced considerably if only the parts that contain calls are extracted and others discarded. Besides the calls, there were several kinds of noise sources present in the recordings.

The most common noise type is a narrowband periodic ticking noise. It can usually be seen at frequency 23 kHz but sometimes also at higher frequencies. It seems to be present in all of the files and caused by the recording device itself. Other common noise type is the sound of water drops hitting a surface and it originates from rain or morning dew. It is broadband noise and it can be seen as vertical lines in the spectrogram (see Fig. 4.3). Besides those most common signals, there are many other types of noise in the high frequencies such as some grasshopper and bird sounds.

4.2 Methods

Median clipping algorithm (Section 3.2.3) was selected as a method for this problem. It is a general method and it does not require any training data. The method needed to be general because the aim is to detect all bat sounds from all species. It also needed to be unsupervised as obtaining a large training dataset was not feasible for this work.

Feature-based methods (Section 3.2.1) were considered but it is unclear which feature would be the best for this purpose. Four features were considered: energy, zero-crossing-rate, spectral flatness and spectral flux. Their suitability was estimated visually

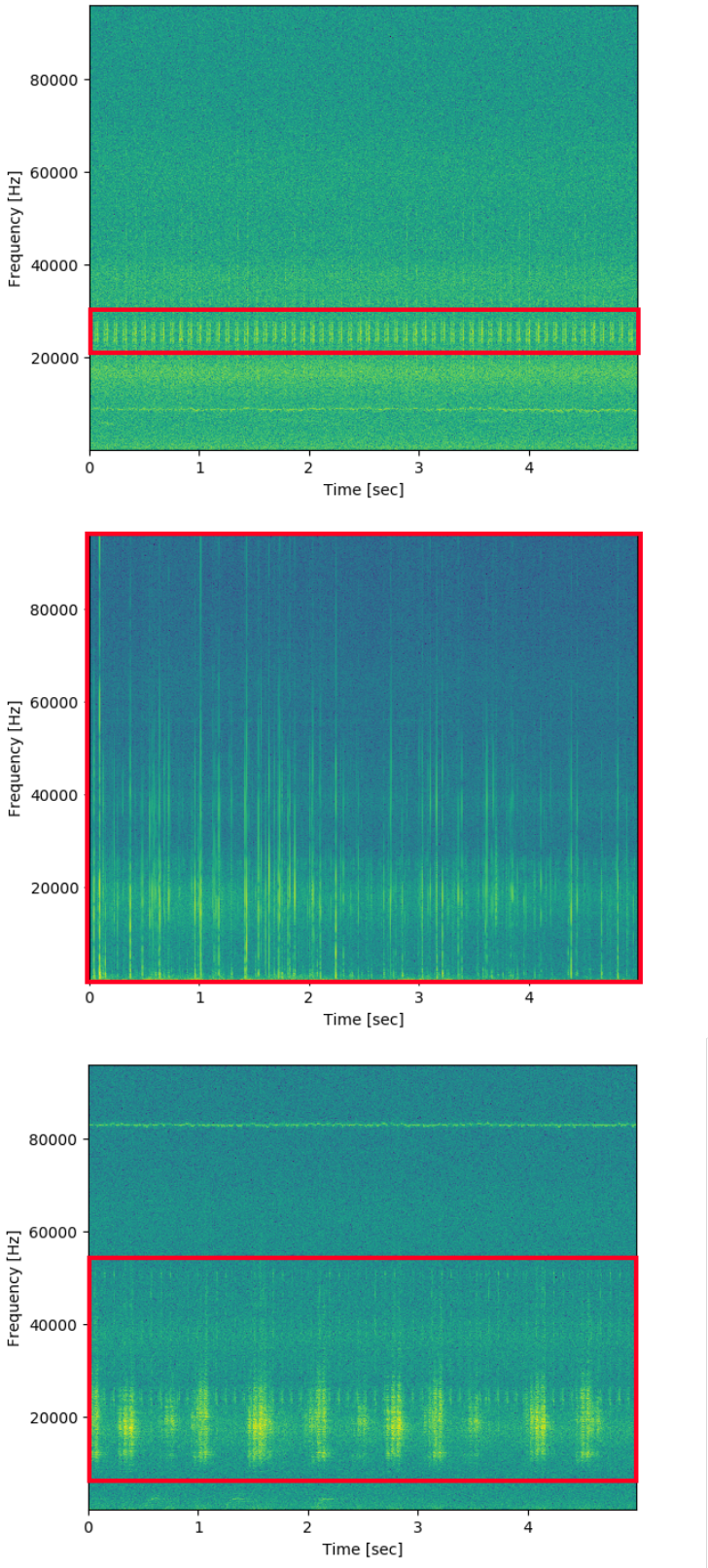

Figure 4.3: Different types of noise in the audio files. From top to bottom: device noise, rain, grasshopper.

by plotting the feature values at different time points under a spectrogram as in the Fig. 3.1. None of these features seemed to separate noise and bat sounds particularly well, and it seemed that some more complex features would be needed.

The detection algorithm has three steps (Fig 4.4). First, a spectrogram is computed. Next, simple noise reduction is applied. After that, comes the actual detection algorithm. Some modifications were made to the detection algorithm to make it work better for bat call detection. These steps are described in more detail in the following subsections.

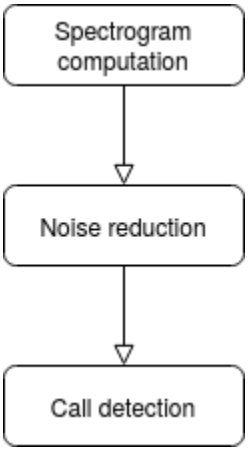

Figure 4.4: Steps of the call detection algorithm.

4.2.1 Spectrogram computation

A magnitude spectrum (see Section 2.3) was computed with following parameters: window size $L = 20$ ms and window step $S = 5$ ms. The overlap between consecutive frames is then 75 %. Hamming window was selected as the window function $w(n)$ (see Section 2.2).

Spectrogram computation has a high memory usage especially when the sampling rate and overlap are high. In order to decrease the risk of running out of memory, it was decided to process the audio in parts. All 10 minute audio recordings were processed in two parts of equal size with 5 second overlap. Overlap was added so that calls near the edges of splitted audio parts would not go unnoticed.

4.2.2 Noise reduction

Spectral noise gating was chosen as a method for reducing Gaussian noise. The method is described in Section 3.1.3. The method uses a part of an audio that contains only noise to estimate mean $\boldsymbol{\mu}_n$ and standard deviation $\boldsymbol{\sigma}_n$ of the noise spectrum. We decided to assume that the frames with lowest 10% energy are Gaussian-noise-only frames and use them for estimation. Let $\boldsymbol{X}_0, ..., \boldsymbol{X}_{L-1}$ be the frames of the lowest 10% energy. L is the

number of frames and K the length of a frame. Then the estimates are calculated for each frequency band k with the common equations for mean and variance:

$$\boldsymbol{\mu}_n(k) = \frac{1}{L}\sum_{l=0}^{L-1} \boldsymbol{X}_l(k)$$

$$\boldsymbol{\sigma}_n(k) = \sqrt{\frac{1}{L}\sum_{l=0}^{L-1} |\boldsymbol{X}_l(k) - \boldsymbol{\mu}_n(k)|^2}$$

4.2.3 Call detection

The call detection algorithm can be divided into the following six steps:

1. Binary mask extraction by median clipping
2. Reducing broadband noise
3. Image processing
4. Filtering out small components
5. Joining close components
6. Getting the final result

The algorithm extracts a binary mask in the first step and then process it in the following steps to get the final result. These steps are shown visually in Figure 4.5. The algorithm differs from the general median clipping algorithm introduced in 3.2.3 in three ways: a modification is added to the binary mask computation and two extra steps (2 and 5) are added. These modifications aim to reduce false detections that are due to noise, especially device and broadband noise.

1. Binary mask extraction by median clipping

First, a binary mask is obtained by a method that is based on a median clipping algorithm (Section 3.2.3). The mask is computed in parts of length 300 frames (1.5 s) as it was observed that that the method works better if the mask is computed in short parts rather than computing it for the whole spectrogram at once (Section 5.1).

The general median clipping method computes a column mask and a row mask and the final mask is combination of these two. In this work, the row mask is left out as it seemed unnecessary (discussed more in Section 5.1).

With the general median clipping algorithm, the AudioMoth device noise would almost always be present in the mask and cause a lot of false positives. Because of that, we made a modification to it. The fact that the device noise is highly periodic, occurring approximately every 17 frame (85 ms), was utilized. In addition to a normal column

mask, a column mask where only every 17th frame is taken into account is computed. Only the pixels with value 1 in both of these masks get the value 1 in the final mask. Figure 4.6 shows how this modifications affects the mask.

Formally, a normal column mask is computed as follows. Let $\boldsymbol{X}_0, ..., \boldsymbol{X}_{L-1}$ be the frames of the magnitude spectrum. In total there are L frames and each frame has a length of K. Then a column mask $\boldsymbol{M}$ is calculated for each frame l and frequency band k as:

$$\boldsymbol{M}_l(k) = \begin{cases} 1, & \text{if } \boldsymbol{X}_l(k) > n * \text{median}[\boldsymbol{Y}_l(k)] \\ 0, & \text{otherwise} \end{cases}, \tag{4.1}$$

where median$[\boldsymbol{Y}_l(k)]$ is a median of a set

$$\boldsymbol{Y}_l(k) = \{\boldsymbol{X}_p(k) | p \in \mathbb{Z} \wedge 0 \leq p < L\}$$

and n is some number ($n = 3$ is commonly used).

A second column mask where only every 17th frame is taken into account is calculated similarly than the normal column mask with equation 4.1 but the set $\boldsymbol{Y}_l(k)$ is a little different:

$$\boldsymbol{Y}_l(k) = \{\boldsymbol{X}_p(k) | p \in \mathbb{Z} \wedge 0 \leq p < L \wedge \exists q \in \mathbb{Z} : p + 17q = l\}.$$

2. Reducing broadband noise

Broadband noise is another noise type that affects the performance of the detector a lot. A simple method was developed in order to reduce at least some of the false positives it causes. It was assumed that if a column has 5 or more pixels with value 1 in both frequency range 0-10 kHz and 10-20 kHz or if there are 2 or more pixels with value 1 in the range 15-16 kHz, then it contain broadband noise. The method is based on the two assumptions:

1. Columns that have a lot of noise in low frequency (< 20 kHz) usually contain only broadband noise and not bat calls

2. Columns that have noise with a high frequency but below the frequency of the bat calls usually contain only noise

After the broadband noise reduction, all pixels outside the frequency range 20-80 kHz are set to zero as bat calls are assumed to be in that range.

3. Image processing

Standard image processing is applied to the mask. First Gaussian blur, then closing and finally median blur is applied (see Section 2.7). A 3x3-sized kernel is used with the blur

operations and a 3x5-sized kernel is used with closing. The focus on the image processing is to reduce most of the noise that is left without reducing the bat calls too much.

4. Filtering out small components

After the mask is processed, an algorithm that finds connected components is applied (see Section 2.7.4). In this case, a pixel is said to be connected to another if it touches it edges or corners (8-connectivity). After extracting these components, they are filtered. Small components usually originate from noise so only components that have area more than 50 pixels, width more than 2 pixels (0.01 s) and height more than 10 pixels (500 Hz) are kept.

5. Joining close components

A connected component approximately corresponds to a pulse in a bat call. In the fifth step of the algorithm, connected components are grouped so that a group corresponds to a call. The components are grouped by their distance from each other. In this case, all components with distance less than 100 pixels (0.5 s) in the x-axis are regarded belonging to same group if the bottom and top part of these components do not differ too much (50 pixels i.e. 2.5 kHz and 200 pixels i.e. 10 kHz used as thresholds). The groups are filtered based on how many components they have. It is assumed that a bat call has at least three pulses so all groups that have less than three components are filtered out.

6. Getting the final result

In the last step, a time range is extracted from the groups. Because the start and end parts of the calls are usually faint and easily gets left out, a method similar to the hangover scheme is applied. First, all components that have a distance in the x-axis less than 500 pixels (2.5 s) are included to the group. Then, a 200 pixel (1 s) padding is added to the result. So if the rightmost frame of the group would be 1000 and leftmost frame 1500, the resulting range would be 800-1700. Finally, the range is converted to seconds.

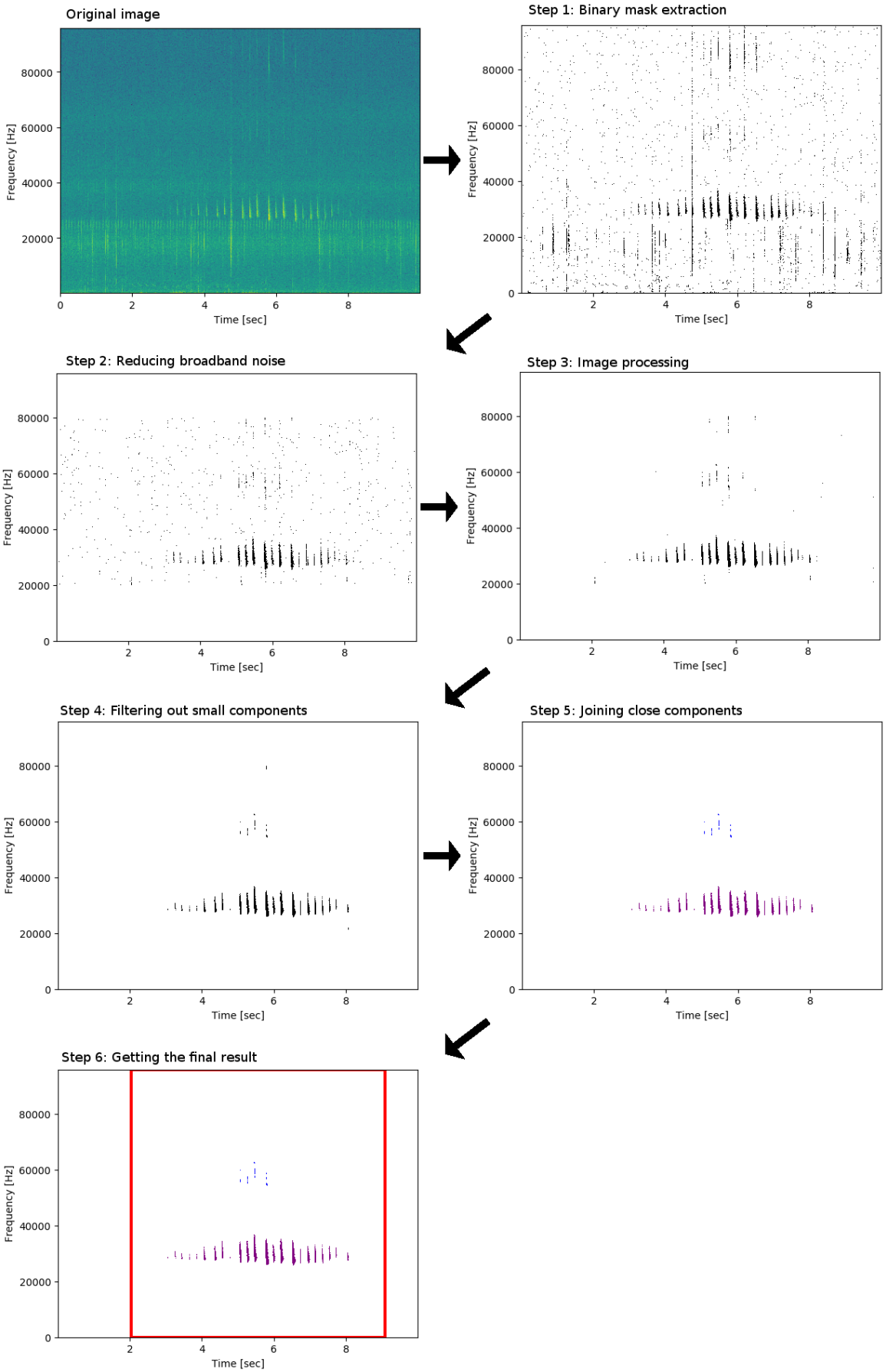

Figure 4.5: The first image shows the starting image which is a spectrogram of a bat call. Stemps 1-6 show how the binary mask looks like in different steps of the call detection algorithm. Steps 5-6 are colored to show how the components are grouped into two groups. The final result is depicted in step 6 with red rectangle.

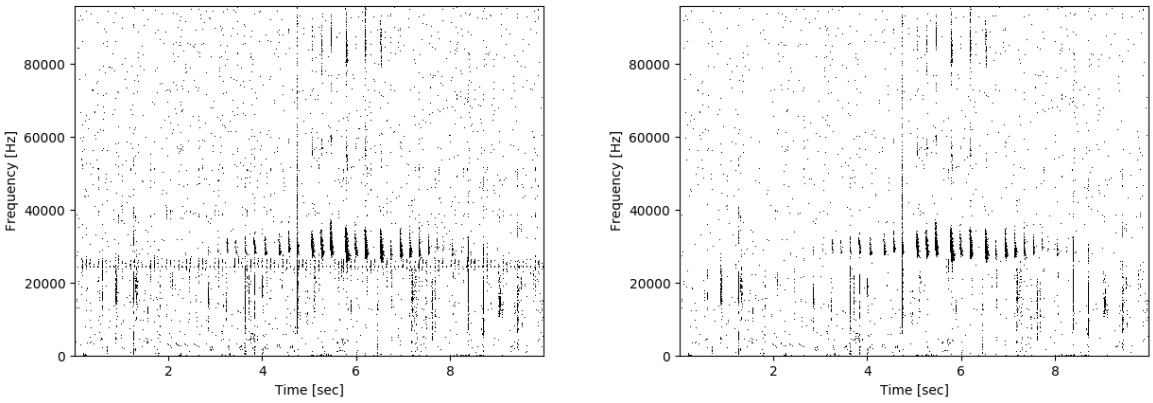

Figure 4.6: The effect of using the second column mask that aims to reduce device noise. On the left mask, device noise can be seen at the frequency 23 kHz. On the right mask, where the second column mask is used, the device noise is not visible.

5. Results

In this section, we evaluate the performance of the detector described in Section 4.2. We start by looking into how different parameters values affected the performance of the detector with the development dataset. Then we evaluate how the detector performed with the testing dataset. Finally, we compare its performance with two existing programs, Kaleidoscope from Wildlife Acoustics [3] and Bat Detective [29].

5.1 Evaluation of detector parameters

The detector has several parameters, such as the spectrogram window size L and median clipping threshold n. The parameters were fine-tuned with the development dataset (see Section 4.1.1). The process was carried out so that only one parameter was changed at a time and others were kept at their optimal values. F2-score (see Section 2.8.1) was used as a performance measure.

For computing the performance measure, manual annotations and detector results were converted to a binary vector of a length of the spectrogram. Value 0 in the vector means that the corresponding spectrogram frame does not belong to bat call and value 1 means that it belongs to bat call.

Table 5.1 contains the results of some of the most important parameters. In the following subsections, we will look into them in more detail. The parameter options that got the best F2-score were chosen for the final version of the detector. They are highlighted in green in the table.

Phase	Parameter	Value	Precision	Recall	F2-score
Spectrogram computation	Window size	10 ms	0.823	0.891	0.876
		20 ms	0.759	0.957	0.910
		28.3 ms	0.613	0.913	0.832
	Overlap	52.8%	0.855	0.842	0.845
		64.6%	0.761	0.911	0.876
		75%	0.759	0.957	0.910
Noise reduction	Remove noise	Yes	0.759	0.957	0.910
		No	0.818	0.838	0.834
Call detection step 1	Use row mask	Yes	0.754	0.958	0.909
		No	0.759	0.957	0.910
	Segment length	1 s	0.789	0.908	0.881
		1.5 s	0.759	0.957	0.910
		2 s	0.742	0.934	0.888
		5 s	0.528	0.924	0.803
		10 s	0.107	0.925	0.366
	Remove clock noise	Yes	0.759	0.957	0.910
		No	0.041	0.965	0.177
Call detection step 2	Remove broadband noise	Yes	0.759	0.957	0.910
		No	0.213	0.950	0.562
Call detection step 5	Minimum number of calls	2	0.444	0.978	0.788
		3	0.759	0.957	0.910
		4	0.790	0.844	0.833
Call detection step 6	Padding	0.5 s	0.853	0.917	0.904
		1 s	0.759	0.957	0.910
		1.5 s	0.697	0.966	0.897

Table 5.1: The performance for some of the most important parameter choices on the validation dataset. Rows with parameters that have the best F2-score are highlighted in green.

5.1.1 Spectrogram computation

Window size L and the overlap which defines the window step S are the most important parameters in the spectrogram computation. The best performance was achieved with window size 20 ms and overlap 75 % (see Table 5.1). It seems that the bigger the overlap is, the better the recall is. Bigger values than 75% were not considered because computing time and memory requirement increase when the overlap gets bigger.

5.1.2 Noise reduction

It seems that it is useful to apply Gaussian noise reduction before the call detection because it improved the performance with the development dataset (see Table 5.1). Recall and F2-score improved quite much. However, precision decreased a little. Noise reduction makes the sounds stand out from the background more which increases the number of bat calls detected but also increases false detections a little.

5.1.3 Call detection step 1

Step 1 of the call detection algorithm has three parameters of interest: a segment length, whether to use the row mask or not and whether to use the mask that reduces device noise or not.

It was observed that the segment length should be quite short in order to get the best result (see Table 5.1). The segment length affects especially precision. The precision improved from 10.7 % to 75.9 % when using a segment length of 1.5 s over 10 s. It seems that short segment length reduces continuous noise that varies little over time quite effectively.

The row mask did not affect the performance much, it was very slightly worse when the row mask was used (see Table 5.1). The original purpose of the row mask is to reduce broadband noise. The reason why it did not work might be the fact that we are using the whole frequency range from 0 to 96 kHz. The amount of noise spanning across the whole frequency range is low.

The mask for reducing the device noise seemed to be very important for this method to work. It improved precision from 4.1% to 75.9% in the development dataset. It might hinder the detection of bat calls that have the same pulse interval than the device but it did not seem to do that too much as the recall did not drop significantly. One reason for that might be that the pulse interval and frequency usually varies a little in a bat call unlike in the device noise.

5.1.4 Call detection step 2

The simple broadband noise reduction technique introduced in Section 4.2.3 improved the precision considerably (from 21.3 % to 75.9%) and did not affect the recall. It seems that it works quite well for its intended purpose.

5.2 Performance

The performance of the detector was determined using the testing dataset (see Section 4.1.1). The same performance measures were used as in the parameter evaluation (see Section 5.1). Results are displayed in Table 5.2. The recall of the detector was quite good (90.6%) but the precision was not as high (47.0%).

The detector did not perform as well with the testing than with the development dataset. Especially the precision fell quite much (from 75.9% to 47.0%). The recall did not change as much. We will go through possible reasons in the next subsection.

Dataset	Precision	Recall	F2-score
Development	0.759	0.957	0.910
Testing	0.470	0.906	0.764

Table 5.2: The performance of the detector.

5.2.1 Performance analysis

One reason for the decrease in performance between the development and the testing dataset is overfitting. Parameters were fine-tuned to give the best result with the development dataset so it is natural that the performance is better on it.

Another reason seems to be that the testing dataset contains more challenging files, even though they were randomly selected. The most common noise types, rain and device noise, are usually either present during the whole file duration or not present at all. Because of that, a single file can have a big effect to the performance. Testing dataset contains few files with challenging noise that affected the precision much.

The phenomenon of datasets having different distributions is called covariate shift [38]. Covariate shift is commonly caused by sample selection bias. In our work, the bias is caused by having data from too few files. It would perhaps have been better to select short clips from many different files rather than longer clips from just a few. There are several ways to detect and handle covariate shifting but they were not utilized as it would have increased the scope of this work considerably.

Fig. 5.1 shows spectrogram clips from the top three files that caused most of false positives with the testing dataset. Most of the false positives (51.5%) were caused by a single file. That file contains broadband noise that seems to originate from dropping water. In this case, unlike in the example in 4.3, the noise does not form clear lines that start from low frequencies to the spectrogram. Instead, it forms lines that seem to start from around 20 kHz. The detector does not work with this kind of noise because it can only reduce broadband noise that starts from lower frequencies.

Another noise type that caused many false positives is the noise that can seen as straight thin horizontal line in the spectrogram and which probably originates from the recording device. It changes little in a frequency over time. Sometimes its frequency fluctuates a little more causing false positives.

False negatives were mostly caused by faint bat calls. There were also some calls that overlapped with grasshopper noise. Fig. 5.2 shows spectrogram clips from the top three files that caused the most of false negatives.

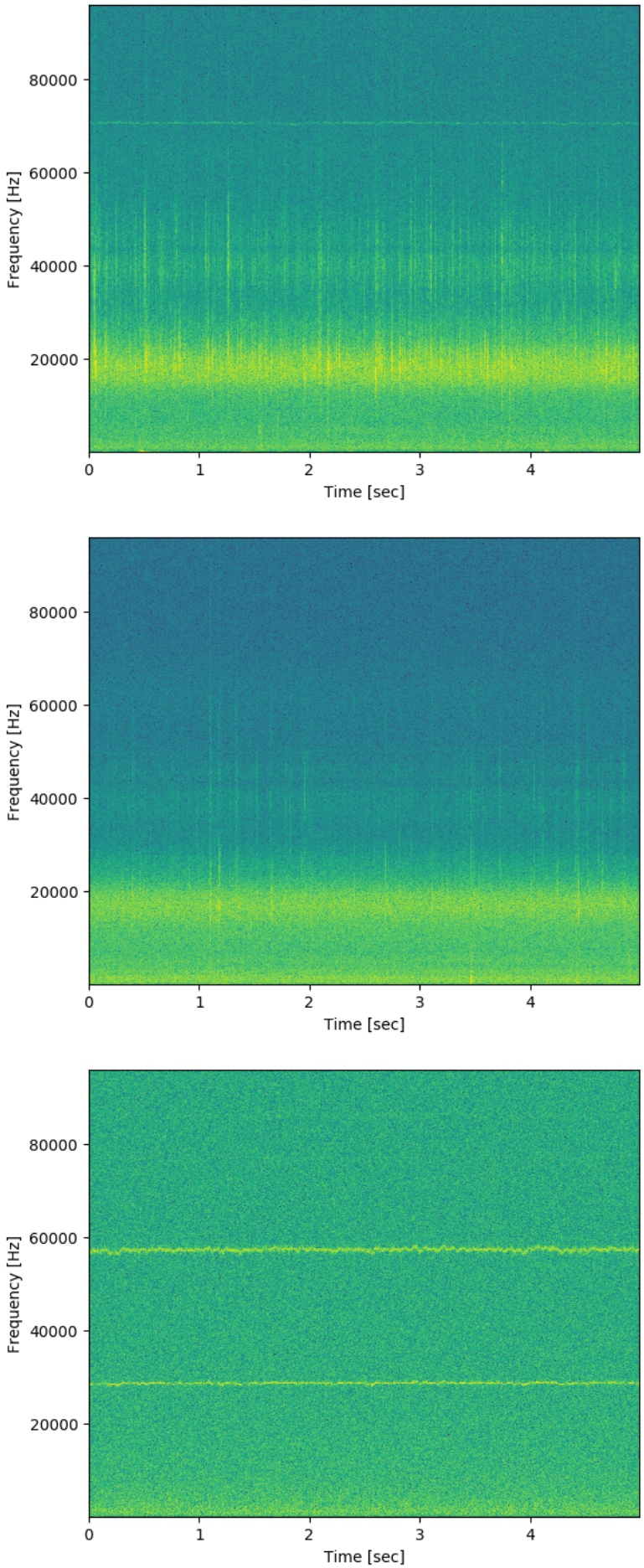

Figure 5.1: Examples from the top three files that caused the most false positives. Top and middle: dropping water. Bottom: noise that probably originates from the device.

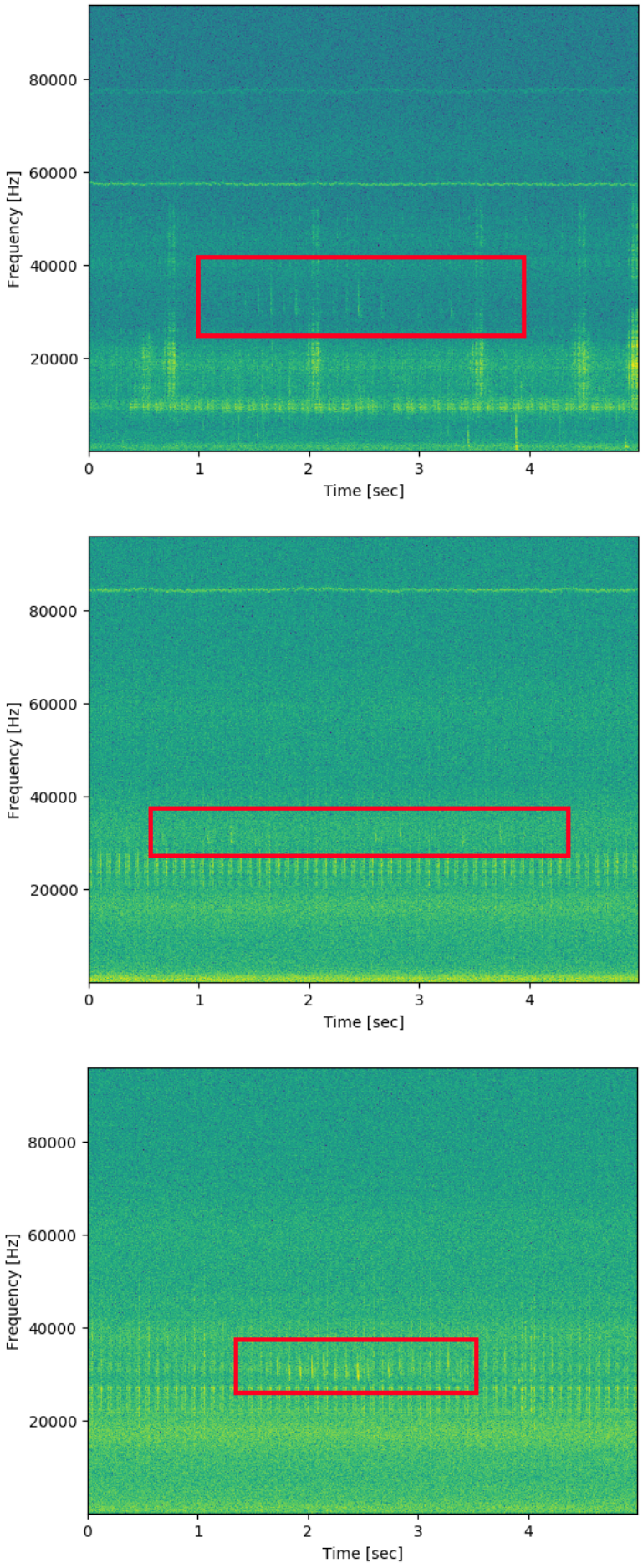

Figure 5.2: Examples from the top three files that caused the most false negatives. The calls are quite difficult to see in the spectrogram as they are quite faint.

5.3 Comparison with other methods

We compared our method with two freely available bat call detection programs, Kaleidoscope (version 5.2.1) [3] and Bat Detective [29].

Kaleidoscope is a commercial program that has two versions, a free version and a pro version. The pro version is paid and has more features such as automatic bat call identification to species. In this comparison, we used the free version as the call detection feature should be the same in both versions. The documentation of Kaleidoscope states that the detection algorithm is based on detecting smooth continuous narrowband frequency sweeps [3].

Bat Detective is an open-source program that is based on a convolutional neural network classifier (see e.g. [26]). Their classifier was trained with data consisting of 2,812 audio clips of length 3.84s that were collected from Romania and Bulgaria [29]. They chose these countries because most of the common European bat species appear there. The clips contained 4,782 bat search-phase echolocation call annotations. They were annotated by a single person who was the most active person in their citizen science annotation project and whose annotations seemed to be of high quality.

All three programs give their results in different formats. Our program gives the start and end spectrogram frame of each bat call. Kaleidoscope splits the recording into segments of some length, such as 10 seconds, and tells for each of these segments whether or not they contain bat sounds. Bat Detective gives the start and end time of each individual pulse. Thus out of these three programs, Bat Detective gives the most fine-grained results and Kaleidoscope the least. To make the comparison possible, all results are converted to Kaleidoscope format (10 s used as the segment length).

Kaleidoscope and Bat Detective both contain parameters that can be changed. We optimized them with the development dataset. Table 5.3 contains the used parameters.

Program	**Parameter**	**Value**
Kaleidoscope	Minimum frequency range	18 kHz
	Maximum frequency range	80 kHz
	Minimum length of detected pulses	0.4 ms
	Maximum length of detected pulses	2 ms
	Maximum inter-syllable gap	500 ms
	Minimum number of pulses	2
Bat Detective	Detection threshold	0.55

Table 5.3: Parameters used with Kaleidoscope and Bat Detective.

5.3.1 Comparison results

Table 5.4 contains results of the comparison. Our program got the best F2-score with both development and testing datasets. With the development dataset, Kaleidoscope got better score than Bat Detective, but with testing dataset it was the other way around.

Dataset	Program	Precision	Recall	F2-score
Development	Our program	0.881	0.913	0.906
	Kaleidoscope	0.642	0.688	0.679
	Bat Detective	0.413	0.623	0.566
Testing	Our program	0.528	0.841	0.752
	Kaleidoscope	0.291	0.634	0.513
	Bat Detective	0.421	0.703	0.620

Table 5.4: The performance of different call detection programs with both the development and the testing dataset

Both our program and Kaleidoscope had worse precision and recall with the testing dataset compared to the development dataset. Especially there is big difference in the precisions. In this regard Bat Detective differs from these two, its precision and recall both were a little better with the testing than with development set.

AudioMoth device noise was a challenge for each of these detectors. All of them had problems with it, but the type of the device noise that caused the most problems seemed to differ between them. The rain noise was also challenging for all of them. Grasshopper noise caused a lot of false positives for Kaleidoscope but not so much for our program and Bat Detective.

5.3.2 Comparison analysis

Our program performed excellently in this comparison, especially considering how Kaleidoscope is a commercial program and Bat Detective is developed by a sizable research group. However, our program had a big advantage over the two in this comparison because it was developed especially for recordings that were recorded in Finland with AudioMoth recording device. Our method might not work so well with data that is recorded with some other recording device or in some other country where the prevalent bat species or environmental noise differs.

Bat Detective would most likely performed better if we would had provided more training data for it. The data that was used to train Bat Detective was recorded with a different recording device than ours which might explain why it had problems with the AudioMoth device noise. It is also unclear how well bat species that appear in Finland

were represented in their training data that was collected from Romania and Bulgaria.

6. Conclusions

We developed a method for detecting bat calls from audio recordings based on the median clipping method. The method was focused on recordings that are recorded with AudioMoth recording device in Finland. The method gave good results and outperformed two existing bat call detection programs by a wide margin in terms at both precision and recall.

We added several modifications to the basic median clipping method. One of them aims to reduce the effect of AudioMoth device noise and one the effect of rain noise. The basic method cannot differentiate between those kinds of noise and bat sounds which made these modifications critical for the success of the method. We also utilized the fact that bat calls usually consist of several consecutive pulses to get better results.

We tested the performance of our method with a dataset that contains short audio recordings from all over Finland that we had annotated. The recordings are real-world field recordings which means that they contain a lot of faint bat calls (bats far away from the microphone) and noise. Even so, our method was able detect almost all bat calls (the recall was over 90 %) and did not make too many false detections.

Bat sound detection may be an easier task than bird song detection since most of the noise is concentrated on low frequencies and does not overlap with bat sounds. It is still very challenging as device, rain and grasshopper noise all proved to be difficult to separate from bat calls. Detecting faint bat calls is also sometimes challenging even for a human.

For our detector, the greatest challenge was the rain noise that caused several false detections. The modification that we added to the basic median clipping method only improved results to a certain extent but it seems that some more efficient method would be needed to increase precision further.

Our detector works well with our data but it is unclear how well it would work with data that has different measuring environment. More work would be required to determine how general this method is and how well it would work with different data.

For future work, it would be important to collect and annotate more data. To our knowledge, currently there are no large freely available annotated bat call datasets. A large dataset would make performance evaluation and comparison more objective. It is

also required for developing more general-purpose methods or methods that are based on supervised machine learning.

Bibliography

[1] Binary acoustic technology. http://binaryacoustictech.com. Accessed: 2020-03-23.

[2] Lukiolaiset lepakkotutkijoina. https://blogs.helsinki.fi/batscience/lukiolaiset_lepakkotutkijoina/. Accessed: 2020-03-30.

[3] Wildlife acoustics. http://wildlifeacoustics.com. Accessed: 2020-09-21.

[4] I. Agranat. Bat species identification from zero crossing and full spectrum echolocation calls using hidden markov models, fisher scores, unsupervised clustering and balanced winnow pairwise classifiers. *The Journal of the Acoustical Society of America*, 133:3311, 05 2013.

[5] J. Alam, P. Kenny, P. Ouellet, T. Stafylakis, and P. Dumouchel. Supervised/unsupervised voice activity detectors for text-dependent speaker recognition on the rsr2015 corpus. In *Proc. of Odyssey 2014: The Speaker and Language Recognition Workshop*, pages 123–130, Joensuu, Finland, 16-19 June 2014.

[6] S. E. Anderson, A. S. Dave, and D. Margoliash. Template-based automatic recognition of birdsong syllables from continuous recordings. *The Journal of the Acoustical Society of America*, 100(2):1209–1219, 1996.

[7] F. Beritelli, S. Casale, G. Ruggeri, and S. Serrano. Performance evaluation and comparison of g.729/amr/fuzzy voice activity detectors. *Signal Processing Letters, IEEE*, 9:85 – 88, 04 2002.

[8] S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2):113–120, April 1979.

[9] E. Çakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[10] U. Camargo, P. Somervuo, and O. Ovaskainen. Protax-sound: A probabilistic framework for automated animal sound identification. *PLOS ONE*, 12:e0184048, 09 2017.

[11] S. Fagerlund and A. Härmä. Parametrization of inharmonic bird sounds for automatic recognition. In *2005 13th European Signal Processing Conference*, pages 1–4, 2005.

[12] M. Fenton and G. Bell. Recognition of species of insectivorous bats by their echolocation calls. *Journal of Mammalogy*, 62:233, 05 1981.

[13] D. K. Freeman, G. Cosier, C. B. Southcott, and I. Boyd. The voice activity detector for the pan-european digital cellular mobile telephone service. In *International Conference on Acoustics, Speech, and Signal Processing,*, pages 369–372 vol.1, May 1989.

[14] G. Fritsch and A. Bruckner. Operator bias in software-aided bat call identification. *Ecology and Evolution*, 4:2703–2713, 07 2014.

[15] T. Giannakopoulos and A. Pikrakis. *Introduction to Audio Analysis: A MATLAB Approach*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 2014.

[16] A. Graps. "an introduction to wavelets". *IEEE Comp. Sci. Engi.*, 2:50–61, 02 1995.

[17] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987.

[18] C. Hawkins (Walters), A. Collen, T. Lucas, K. Mroz, C. Sayer, and K. Jones. *Challenges of Using Bioacoustics to Globally Monitor Bats*, pages 479–499. Springer, New York, NY, 03 2013.

[19] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25 – 43, 2017.

[20] A. Henríquez, J. B. Alonso, C. M. Travieso, B. Rodríguez-Herrera, F. B. nos, P. Alpízar, K. L. de Ipiña, and P. Henríquez. An automatic acoustic bat identification system based on the audible spectrum. *Expert Systems with Applications*, 41(11):5451 – 5465, Feb. 2014.

[21] P. Hill. *Audio and Speech Processing with MATLAB*. CRC Press, Inc., 1st edition, 12 2018.

[22] J. D. Johnston. Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*, 6(2):314–323, 1988.

[23] G. Jones, D. Jacobs, T. Kunz, and P. Racey. Carpe noctem: The importance of bats as bioindicators. *Endangered Species Research*, 8:93–115, 07 2009.

[24] D. M. Kiapuchinski, C. R. E. Lima, and C. A. A. Kaestner. Spectral noise gate technique applied to birdsong preprocessing on embedded unit. In *2012 IEEE International Symposium on Multimedia*, pages 24–27, 2012.

[25] B. Krause. Anatomy of the soundscape: Evolving perspectives. *Journal of the Audio Engineering Society. Audio Engineering Society*, 56:73 – 80, 01 2008.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[27] M. Lasseck. Bird song classification in field recordings: Winning solution for nips4b 2013 competition. In *Workshop on Neural Information Processing Scaled for Bioacoustics*, pages 176–181, 2013.

[28] M. Lasseck. Large-scale identification of birds in audio recordings. In *CLEF*, 2014.

[29] O. Mac Aodha, R. Gibb, K. E. Barlow, E. Browning, M. Firman, R. Freeman, B. Harder, L. Kinsey, G. R. Mead, S. E. Newson, I. Pandourski, S. Parsons, J. Russ, A. Szodoray-Paradi, F. Szodoray-Paradi, E. Tilova, M. Girolami, G. Brostow, and K. E. Jones. Bat detective—deep learning tools for bat acoustic signal detection. *PLOS Computational Biology*, 14(3):1–19, 03 2018.

[30] D. Mellinger and C. Clark. Recognizing transient low-frequency whale sounds by spectrogram correlation. *The Journal of the Acoustical Society of America*, 107:3518–3529, 07 2000.

[31] R. Mudhar. Recordings of ultrasonic vocalisations of bats. https://www.wildlife-sound.org/resources/equipment/2-uncategorised/233-recordings-of-ultrasonic-vocalisations-of-bats. Accessed: 2020-03-16.

[32] K. Murray, E. Britzke, and L. Robbins. Variation in search-phase calls of bats. *Journal of Mammalogy - J MAMMAL*, 82:728–737, 08 2001.

[33] M. Myllykoski. *On GPU-accelerated fast direct solvers and their applications in image denoising*. PhD thesis, UmeåUniversity, 08 2015.

[34] S. Parsons and G. Jones. Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. *Journal of Experimental Biology*, 203(17):2641–2656, 2000.

[35] I. Potamitis, S. Ntalampiras, O. Jahn, and K. Riede. Automatic bird sound detection in long real-field recordings: Applications and tools. *Applied Acoustics*, 80:1 – 9, 2014.

[36] N. Priyadarshani, S. Marsland, and I. Castro. Automated birdsong recognition in complex acoustic environments: a review. *Journal of Avian Biology*, 49(5):jav–01447, 2018.

[37] N. Priyadarshani, S. Marsland, I. Castro, and A. Punchihewa. Birdsong denoising using wavelets. *PLOS ONE*, 11(1):1–26, 01 2016.

[38] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[39] L. R. Rabiner and M. R. Sambur. An algorithm for determining the endpoints of isolated utterances. *The Bell System Technical Journal*, 54(2):297–315, Feb 1975.

[40] B. R.G, K. S., A. B., and B. Barkana. *Voiced/Unvoiced Decision for Speech Signals Based on Zero-Crossing Rate and Energy*, pages 279–282. Springer, 01 2010.

[41] S. Rosen. *Signals and Systems for Speech and Hearing*, page 163. BRILL, 2nd edition, 2011.

[42] T. Sainburg. Noise reduction using spectral gating in python. https://timsainburg.com/noise-reduction-python.html. Accessed: 2020-10-20.

[43] J. Saunders. Real-time discrimination of broadcast speech/music. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 2, pages 993–996 vol. 2, 1996.

[44] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1331–1334 vol.2, 1997.

[45] A. Selin, J. Turunen, and J. Tanttu. Wavelets in recognition of bird sounds. *EURASIP J. Adv. Sig. Proc.*, 2007, 01 2007.

[46] D. Stowell, Y. Stylianou, M. Wood, H. Pamula, and H. Glotin. Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge. *CoRR*, abs/1807.05812, 2018.

[47] Z.-H. Tan, A. kr. Sarkar, and N. Dehak. rvad: An unsupervised segment-based robust voice activity detection method. *Computer Speech & Language*, 59:1 – 21, Jan. 2020.

[48] S. Vaseghi. *Least Square Error Wiener-Kolmogorov Filters*, chapter 6, pages 173–191. John Wiley & Sons, Ltd, 2009.

[49] S. M. Williams, S. J. Wolbert, and H. P. Whidden. Evaluation of the program scan'r for sorting ultrasonic recordings of bat vocalizations. In *Proceedings of the North east Bat Working Group, North Branch, NJ*, 2007.

[50] Z. Zhao, S. hua Zhang, Z. yong Xu, K. Bellisario, N. hua Dai, H. Omrani, and B. C. Pijanowski. Automated bird acoustic event detection and robust species classification. *Ecological Informatics*, 39:99 – 108, 2017.